Bulletin of Networking, Computing, Systems, and Software – www.bncss.org, ISSN 2186–5140
Volume 5, Number 1, pages 68–72, January 2016

Friend Recommendation by Using Skyline Query and Location Information

Shufeng Zheng, Asif Zaman, and Yasuhiko Morimoto
Graduate School of Engineering, Hiroshima University
Kagamiyama 1-7-1, Higashi-Hiroshima 739-8521, Japan
Email:{m141458@, d140094@, morimoto@mis.}hiroshima-u.ac.jp

***Abstract*—By leveraging the capabilities of modern GPS-equipped mobile devices, the interest in developing advanced services that combine location-based services with social networking services is growing drastically. This type of social network that combined with location information and social relations is called as location-based social network. Many location-based social network services have functions for finding/recommending friends to a user. Facebook, for example, uses friends-of-friends information to recommend new friends to users. The idea is based on an observation that a person prefers a friend of their friends rather than a random person. However, this approach does not consider the distance and friend influence between user and new recommended friend. Therefore, we proposed a method of friend recommendation that combines of skyline query and check-in information to find promising friends. We conducted a set of experiments to show the quality and accuracy of friend recommendation of our propose approach.**

Keywords—Location-based Social Networks; distance; friend influence; skyline query.

I. Introduction

With the rapid development of mobile devices, global position system (GPS), and Web 2.0 technologies, location-based social networks (LBSNs) have attracted millions of users to share rich information, such as experiences and tips. Social network is not only helping user to maintain their social relationships, but also helps user to find interesting friends. Therefore, potential friends' recommendation function is necessary for each social networking site, and this function should be accurate. Moreover, friend recommendations in LBSNs is a promising and interesting research problem, in which we need to consider social links between users and interactions between users and locations in the recommendation process. We probably know that traditional social networking sites, such as Facebook, always provides us some information to add some new friends. The recommendation of new friends mostly based on friends of friends information. But this idea does not consider the location nor similar interest between user and new recommended friend. For example, a user is more likely to be friends with other users those who share the same geographical location. It is also true in our real life; usually we are connected to people which has stronger connections may than others. For example, if a stranger recommends a new friend to you, you have small chance to accept him/her as your new friend. However, if a familiar friend recommends a new friend to you, she/he would have greater chance to be accepted by you. This is because, we have better trust with our familiar friends. Therefore, we need to consider the friend influence for getting the promising friend to user. In this work, we choose the friends of our familiar friends as candidate friends. We consider the conventional common friend relationship as well as geographical location influence. We recommend new friends based on the similarity score between user and potential candidate's friends.

Following example demonstrates the idea of recommending new friends in location based on social network.

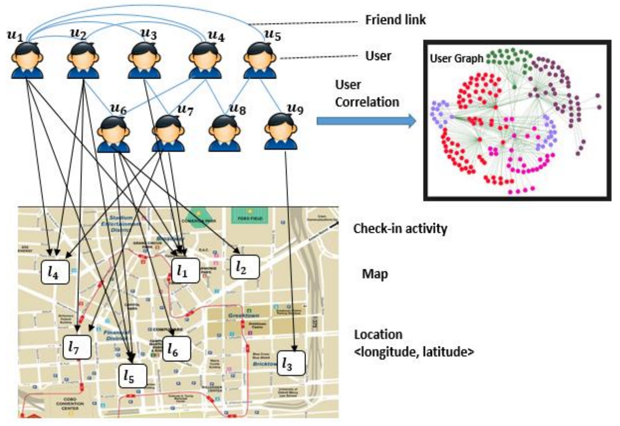

Figure1: Graph representation of user-user friendship and user check-in activity in a LBSN

In Figure 1, the users in LBSN are denoted as U_1, U_2, … , U_9. They are interconnected by social links to form user social networks. The locations are denoted as l_1, l_2, …, l_7, connected with users based on their "check-in" activities. The check-in activities reflect that the users have visited in various locations. Table 1 represents users, candidate friends, common friends which are both friends for the users and their candidate friends, the distance between users and their candidate friends, who are visit the same location, and similarity score. The visit same location shows the same locations that both the user and user's candidate friend have come, and similarity score shows similarity score between users and their candidate friends. If the user and candidate friend visited the same location then they have location similarity. Table 2 represents the user, the user's friend, and the friend influence shows that the familiarity between the user and the user's friend.

User	Candidate friend	Common friend	Distance (km)	Visit Same Location	Similarity Score
U_1	U_6	U_2, U_4	18	l_1, l_5	0.41857
U_1	U_7	U_3, U_4	24	l_1, l_4	0.21252
U_1	U_8	U_4, U_5	1346	Φ	0
U_1	U_9	U_5	31	Φ	0

Table1: User and Candidate friend Example

User	Friend	Friend Influence
U_1	U_2	0.475
U_1	U_3	0.297
U_1	U_4	0.025
U_1	U_5	0

Table2: Friend Influence Example

From Table 1 we can see that if we use the common friend list to recommend new friend to U_1, we will choose U_6, U_7, or U_8 as the friend recommendation. However, if we intensively look at the distance and location similarity then we could discover that U_8 is not a good friend recommendation, compared to remaining candidate (U_6, U_7). Therefore we will choose U_7 and U_6. But in real life, if a very familiar friend recommends a new friend, then the chance of the friend's acceptance is very large. From the column of visit same location in Table 1, we know that U_1 and U_7 visited two same locations, and U_1 and U_6 also visited two same locations. But from Figure 1 and Table 1 we can see that U_6 is friend of U_2 and U_4, and U_7 is friend of U_3 and U_4. As U_6 and U_7 has common friend U_4, so in here, we don't consider U_4. From Table 2 we can see that, U_2 is more familiar than U_3 with respect to U_1. From Table 1, U_6 is closer than U_7 , so U_6 is better than U_7 as a new friend of U_1.

Based on the above example, we want to recommend promising friends that consider of three factors: (a) common friend, (b) distance influence, and (c) similarity score, which is calculate from location similarity and friend influence between user and candidate friends. Among lot of candidates, we have to choose promising friends. We consider skyline query to solve this problem. Skyline query is a technique to select better candidate than others.

The proposed method work as follows:

- Collect the candidate friends and common friend's number based on social relationship link.
- Calculate distances between user and candidate friend based on user check-in information.
- Calculate the location similarity and friend's influence based on check-in information and social relationship link. We use location similarity and friend influence to get the similarity score.
- Finally, using the skyline query method to get friend recommendation list.

The rest of this paper is organized as follows. Section 2 reviews related work. Section 3 presents the notions and properties of friend recommendation by using skyline query. Section 4 shows the experiment and Section 5 is the conclusion

II. RELATED WORK

A. Recommendation system

In recommendation system, there are two widely adopted techniques for recommendation. One is Content-based, and another is collaborative filtering. A content-based system is mainly to choose items based on the correlation and the users' preferences [1]. For collaborative filtering systems, it can be divided into two categories. One is memory-based, and another one is model-based. In the memory based systems [2]. We can use some similarity measure to calculate the similarity between all users, such as the cosine similarity or the Pearson correlation score. Then using the similarity, we can find some similarity items to recommend to user. On the other hand, in model-based filtering systems, the system assume that we can use user similar behavior in rating of items to build up the clusters, Then using the clusters, we can do recommendation.

Under the context of social networking systems, social friendship is shown very useful for collaborative filtering based on recommendation systems, e.g., memory-based [3]. The work reflect that the social friends tend to share common interests and thus we should consider this in recommendation system. On the other hand, the model-based systems has also been explored social friendship [4]. The work is mainly for conventional recommendation systems for recommendation.

B. Local social network

Advances in location-acquisition and wireless communication technologies , the people can add the location information to traditional social networks, a number of location-based social networking services (or LBSNs) has is growing drastically, e.g., Foursquare, Gowalla, and so on. Users can easily share him/her experiences about locations with him/her friends just through the apps on mobile phone. User check-in information is an important information to reflect user behavior. There are have a lot of research in location information to analyze user behavior [5, 6, 7, 8]. Among them, [5, 6, 7, 8] are mainly using GPS datasets to analyze user behavior, but it don't consider user social relationship . [9] is mainly using location- based social networking to analyze user behavior, using user check-in time to analyze user behavior, what time user like to check-in and which location is interesting for user. [10] is mainly through GPS logs to calculate the user similarity by user active track . [11] is mainly through collaborative filtering and geographical influence for location recommendation. Usually friend recommendation based on location information just consider the similarity about check-in same location between two users. In this paper we not just consider similarity but also consider common friend and distance influence for friend recommendation.

C. Skyline Queries Processing

The skyline operator was introduced in [12]. Additionally, S. Borzsonyi et al. propose block-nested-loops (BNL) processing and an extended divide-and-conquer (D&C), and B-tree-based schemes to process results for their new method. In particular, BNL maintains a self-organized window for currently incomparable tuples, to compare with every incoming tuple.

D&C partitions the entire data set into multiple subsets from which local skyline results are computed and then merged. These two algorithms, BNL and D&C, set up a basic foundation for skyline computing and have had a significant impact on later works [13, 14, 15]. Since then, skyline processing has attracted considerable attention in the database community. Kossmann et al. proposed an algorithm to obtain the skyline based on a nearest neighbor approach, which uses a D&C scheme for data indexed by an R-tree [16]. Papadias et al. [17] proposed a Branchand-Bound Skyline (BBS) method based on the best-first nearest neighbor algorithm. Dellis and Seeger [18] proposed a reverse skyline query, which obtains those objects that have the query point as skyline, where each attribute is defined as the absolute difference from objects to query point along each dimension. [19] developed Skyline Space Partitioning approach to compute skylines on a tree-structured P2P platform BATON. In application, the skyline query has already been some work considering the application in a setting including road networks [20].

Skyline query is an approach to select a limited number of “good” items from a large database, and there are many proposals for implementing this approach. In this paper we choose skyline query to choose the candidate friend for user.

III. Problem Definition

In the following we will discuss some properties and definitions of LBSNs friend recommendation skyline query. Furthermore, we also discuss the method for calculating common friend and distance influence and similarity between user and recommendation candidate friends.

A. *Definition 1- LBSNs friend recommendation skyline query (LFRSQ):*

Let U denotes candidate friend list. Furthermore, let $dist_{geo}$ denotes the distance between user and candidate friend, let CF_{num} denotes the common friend number between user and candidate friend, and let UC_{sc} denotes similarity score between user and candidate friend.

For two candidate friend $u_i$and u_j in U, candidate u_i is said to dominate the candidate friend u_j, if u_i is better or equal to in all three dimensions (CF_{num} , $dist_{geo}$, UC_{sc}) than u_j and at least better in one dimension. In our case, for dimension CF_{num} and UC_{sc} higher is better and for dimension $dist_{geo}$ lower is better.

From Figure 1 and table 1. The U_1’s candidate friend U_6 has CF_{num} (U_6) =2, $dist_{geo}$(U_6) = 18 and UC_{sc}(U_6)= 0.475. Another Candidate friend U_7 has CF_{num} (U_7) =2, $dist_{geo}$(U_7) =24 and UC_{sc}(U_7)= 0.21252.we can see$dist_{geo}$(U_6)= 1$<$$dist_{geo}$($U_7$) =24, CF_{num} (U_6)= CF_{num} (U_7)=2 and UC_{sc} (U_6)= 0.475 $>$ UC_{sc}(U_7)= 0.21252. So, we can see U_6 is better than U_7 as a new friend of U_1.

In the next section, we introduce how to get these.

B. *Common Friend Number*

Let U_S denotes a user who is recommended friend by recommendation system, F_k denotes the U_S friend set and U denotes the U_S candidate friend list, FF_k denotes the user friend of friend, and Let CL_F denote the $u_i(u_i \in$ U) friend list. If u_i belongs to the U, u_i is friend of F_k, but not friend of U_S. We define as follows:

$$\text{U} = u_i \in FF_k \land u_i \notin F_k.$$

The CF_{num} is defined as follows:

$$CF_{num} = F_k \land CL_F$$

According this equation, we can get common friend number between user and candidate friend.

C. *Distance Similarity*

We want to know the distance between U_S and candidate friend list U. but as user home locations are not explicitly given, we infer them by user’s check-in information and assume the home location. In here, we choose the user top-3 check-in frequent information and use the gravity center.

For the distance, instead of using the common Euclidean distance, we decided to use the geodetic distance which is the shortest distance between two points on Earth along the Earth’s surface (simplified as a sphere).The distance between U_S and candidate friend u_i ($u_i \in$ U) is defined as follows:

$$dist_{geo}(u_{U_S}^{lat}, u_{U_S}^{long}, u_i^{lat}, u_i^{long}) = \text{r}\cdot\arccos(\sin(u_{U_S}^{lat})\cdot\sin(u_i^{lat}) + \cos(u_{U_S}^{lat})\cdot\cos(u_i^{lat})\cdot\cos(u_{U_S}^{long} - u_i^{long}))$$

Where, r is the Earth’s radius (approx. 6371km).

Assume that, we get U_S’s Latitude 39.94 and Longitude 117.30, candidate friend u_i’s Latitude 39.96 and Longitude 116.45. Now we can use the above defined equation for getting the distance as follows:

$$dist_{geo}(u_{U_S}^{lat}, u_{U_S}^{long}, u_i^{lat}, u_i^{long}) = \text{r}\cdot\arccos(\sin(39.94)\cdot\sin(39.96) + \cos(39.94)\cdot\cos(39.96)\cdot\cos(117.30-116.45)) = 72444.81551 \text{ (m)}.$$

D. *Similarity Score*

1) *Location similarity*

Based on collaborative filtering, we can find users’ implicit preference through aggregating the behaviors of similar users. Let U_S denotes a user who recommended friend by recommendation system. L denotes a set of locations. The check-in activity $c_{i,j}$ means the user u_i has been to the location l_j ϵL. We denote the value of $c_{i,j}$ as follows:

$$c_{i,j} = \begin{cases} 1 & if\ checked-in \\ 0 & if\ not\ checked-in \end{cases}$$

To compute the similarity between U_S and u_i ∈U, in our paper, we used cosine similarity to compute . The similarity $S_{u,i}$ between U_S and u_i is defined as follows:

$$S_{u,i} = \frac{\sum_{l_j \epsilon \text{L}} c_{U_S,j} c_{i,j}}{\sqrt{\sum_{l_j \epsilon \text{L}} {c_{U_S,j}}^2} \sqrt{\sum_{l_j \epsilon \text{L}} {c_{i,j}}^2}}$$

For example, let’s consider $S_{u,i}$ that have shown in Figure 1, Table 1. U_1 has visited locations l_1, l_4, l_5. The candidate friend U_6 of U_1 has visited locations l_1, l_2, l_5, l_6. We can see that U_1 and U_6 has visited same locations l_1, l_5. So, $\sum_{l_j \epsilon \text{L}} c_{U_1,j} c_{U_6,j}$ =2,

$\sqrt{\sum_{l_j \in L} c_{U_1,j}^2} = \sqrt{3}$, $\sqrt{\sum_{l_j \in L} c_{U_6 j}^2} = \sqrt{4}$, hence, we can get the $S_{U_1,U_6} = 0.577$.

2) Friend Influence

In real life, friends tend to have similar behavior and similar interesting, as friends, they might share a lot of common interests. On the other hand, we know friends who have closer social tie may have better trust in terms than others. So, if two friends have more similar check-in behavior, they should have more common interesting , for recommendation friend of friend, the close friend of one person has a large influence for his/her, in here we consider if we want to recommend good friend, we must consider user and user friends social connections. We define this equation based on both of their social connections and similarity of their check-in activities. The friend influence UF_{inf} between U_S and U_S friend u_i is defined as follows:

$$UF_{inf_{u_k}} = \alpha \frac{|F_{U_S} \cap F_{u_k}|}{|F_{U_S} \cup F_{u_k}|} + (1-\alpha) \frac{|L_{U_S} \cap L_{u_k}|}{|L_{U_S} \cup L_{u_k}|}$$

where α is a tuning parameter ranging within[0,1]. F_{U_S} and L_{U_S} denote the friend set and location set of user U_S , and F_{u_k} and L_{u_k} denote the friend set and location set of user U_S friend u_k.

For example, consider UF_{inf} that have shown in Figure 1, Table 1 and Table 2. The friend set of U_1 are U_2, U_3, U_4, U_5, and its visited location set are l_1 , l_4, l_5. We can see, U_1 's friend U_2 has friends U_4 & U_6, and U_2 visited locations are l_4, l_5, l_7. U_1 and U_2 has only one common friend U_4 , so $|F_{U_1} \cap F_{U_2}|$=1 and $|F_{U_1} \cup F_{U_2}|$=4. According the Figure 1, U_1 and U_2 have visited two same locations, so $|L_{U_1} \cap L_{U_2}|$ =2 and $|L_{U_1} \cup L_{U_2}|$ =4. Assume we choose α =0.1, so $UF_{inf_{U_2}} = 0.1 \cdot \frac{1}{4} + 0.9 \cdot \frac{1}{2} =$ 0.475.

Next, we finally define the Similarity Score UC_{sc} by combining the above two aspects. Consider candidate friend u_i have different common friends with U_S, we should consider every friend influence for candidate friend u_i. So, if u_i is friend of u_k the $UF_{inf_{u_k}} = UF_{inf_{u_k}}$ value, If u_i is not friend of u_k the $UF_{inf_{u_k}}$ = 0. The Similarity Score of the candidate friend u_i is defined as follows:

$$UC_{SC_i} = \sum_{u_k \in F_k} UF_{inf_{u_k}} \cdot S_{u,i}$$

where $S_{u,i}$ denote the location similarity between U_S and candidate friend u_i, F_k denotes the U_S friend set ,and $UF_{inf_{u_k}}$ denote the friend influence between U_S and friend u_k.

For example, consider UC_{sc} that have shown in Figure 1, Table 1 and Table 2. U_1 has friends U_2, U_3, U_4 and U_5. U_6 is a candidate friend of U_1. U_6 has U_2 and U_4 as its friend set. From Table2, the friend influence of U_2 = 0.475 and U_4 = 0.025 and the location similarity $S_{u,6}$ between U_1 and U_6 is 0.57735. Based on above value, we can calculate $UC_{SC_6} = UF_{inf_{U_2}} \cdot S_{u,6}$ $+UF_{inf_{U_3}} \cdot S_{u,6} + UF_{inf_{U_4}} \cdot S_{u,6} + UF_{inf_{U_5}} \cdot S_{u,6}$=0+ 0.57735 · 0.475+0+0.57735 · 0.025=0.41857.

IV. Experiment

We use the real-world check-in spots dataset to make an offline experiment to evaluate the quality of the recommendations of our framework.

1) Data Sample

Gowalla is a location-based social networking website where users share their locations by checking-in. The friendship network is undirected and was collected using their public API, and consists of 196,591 nodes and 950,327 edges. We have collected a total of 6,442,890 check-ins of these users over the period of Feb. 2009 - Oct. 2010. The table 3 check-in data. The table for social connection data.

User	Check-in time	Latitude	Longitude	Location-id
4260	2010-08-19T02:09:26Z	37.78147	-122.399	32074
4260	2010-08-15T02:06:27Z	37.78422	-122.403	36242
4260	2010-08-12T02:42:37Z	37.78147	-122.399	32074
4260	2010-08-07T03:49:28Z	37.77199	-122.431	18060

Table3 check-in data

User	Fiend-edge	Type
8	66	Undirected
8	295	Undirected
8	4137	Undirected
8	4234	Undirected

Table4 social connection data

2) Evaluation

In our experiment, we randomly choose a user from the dataset, according the user Check-in information and Social connection information using our method (LFRSQ) to compare the common friend method (CFM) and traditional method(TM). Traditional method(TM) is just consider same location between user and candidate friend. We choose the CFM and TM recommendation size equal LFRSQ size.

Function	Average Common friend	Average distance(km)	Average Similarity score
LFRSQ	3.6	103	0.02385
CFM	4.4	670	0.00653
TM	1.8	95	0.01103

Table5 Compare Function

From table 5, compare with the CFM we can find LFRSQ has a better performance in Average distance and Average Similarity score. Compare with the TM, we can find our method is better in Average Common friend and Average Similarity score, in the Average distance The TM has a better result but not significant.

V. Conclusion

In this paper, we proposed a LBSNs friend recommendation skyline query. In our method, we do not just consider using common friend number to recommend new friend, but also consider the user's check-in information to get the distance and similarity for friend recommendation, finally we used LBSNs friend recommendation skyline query to search optimal friend for the user. Based on experiment, we can see our method is better than other method to recommend new friend.

Acknowledgment

This work is supported by KAKENHI (23500180, 25.03040) Japan. A. Zaman is supported by Japanese Government MEXT Scholarship.

References

[1] A. M. Ferman, J. H. Errico, P. van Beek, and M. I. Sezan. Content-based filtering and personalization using structured metadata. In JCDL, page 393, 2002.

[2] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In SIGIR '99: Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pages 230–237, New York, NY, USA, 1999. ACM.

[3] M. Jamali and M. Ester. TrustWalker: a random walk model for combining trust-based and item-based recommendation. In KDD, pages 397–406, 2009.

[4] H. Ma, I. King, and M. R. Lyu. Learning to recommend with social trust ensemble. In SIGIR, pages 203–210, 2009.

[5] V. W. Zheng, Y. Zheng, X. Xie, and Q. Yang. Collaborative location and activity recommendations with gps history data. In WWW, pages 1029–1038, 2010.

[6] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining interesting locations and travel sequences from gps trajectories. In WWW, pages 791–800, 2009.

[7] X. Cao, G. Cong, and C. S. Jensen. Mining significant semantic locations from gps data. PVLDB, 3(1):1009–1020, 2010.

[8] V. W. Zheng, B. Cao, Y. Zheng, X. Xie, and Q. Yang. Collaborative filtering meets mobile recommendation: A user-centered approach. In AAAI, 2010.

[9] S. Scellato, A. Noulas, R. Lambiotte, and C. Mascolo. Socio-spatial properties of online location-based social networks. In ICWSM '11, 2011.

[10] K. W.-T. Leung, D. L. Lee, and W.-C. Lee. CLR: a collaborative location recommendation framework based on co-clustering. In SIGIR, 2012.

[11] Ye, M.; Yin, P.; Lee, W.-C.; and Lee, D.-L. 2011. Exploiting geographical influence for collaborative point-of-interest recommendation. In Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval, 325–334. ACM.

[12] S. Borzsonyi, D. Kossmann, K. Stocker. The Skyline Operator. In ICDE 2001

[13] Chan, C.Y., Jagadish, H.V., Tan, K.-L., Tung, A.K.H., Zhang, Z.: On high dimensional skylines. In: Proceedings of EDBT Conference, pp. 478–495 (2006)

[14] Godfrey, P., Shipley, R., Gryz, J.: Maximal vector computation in large data sets. In: Proceedings of VLDB Conference, pp. 229–240 (2005)

[15] Lin, X., Yuan, Y., Wang, W., Lu, H.: Stabbing the sky: efficient skyline computation over sliding windows. In: Proceedings of ICDE Conference, pp. 502–513 (2005)

[16] Kossmann, D. Ramsak, F.: Shooting stars in the sky: an online algorithm for skyline queries. In: International Conference on Very Large Data Base (VLDB), pp. 275–286 (2002)

[17] D. Papadias, Y. Tao, G. Fu, and B. Seeger. An optimal and progressive algorithm for skyline queries. In Proc. SIGMOD, pages 467-478. ACM, 2003.

[18] E. Dellis and B. Seeger, "Efficient computation of reverse skyline queries," in Proceeding of the 33rd International Conference on Very Large Data Bases, 2007, pp. 291-302.

[19] Shiyuan Wang, Beng chin Ooi, Anthony Tung, Lizhen Xu, "Efficient Skyline Processing on Peer to Peer Networks", IEEE Int" l Conf. Data Eng. (ICDE), 2007.

[20] S.M. Jang and J.S. Yoo. Processing continuous skyline queries in road networks. In Proc.CSA, page 353-356. IEEE,2008.